Classification and Persistence Analysis of Tie Strength on GitHub

Ana Flávia C. Moura
Universidade Federal de Minas Gerais
anaciriaco@dcc.ufmg.br

Gabriel P. Oliveira
Universidade Federal de Minas Gerais
gabrielpoliveira@dcc.ufmg.br

Michele A. Brandão
Instituto Federal de Minas Gerais
michele.brandao@ifmg.edu.br

Mirella M. Moro
Universidade Federal de Minas Gerais
mirella@dcc.ufmg.br

ABSTRACT

Relationships between users in social networks are evaluated in different ways. Here, our goal is to measure the strength of the ties between GitHub users by considering the temporal aspect of the network. Specifically, we analyze the evolution of these relationships by applying a classification algorithm over a GitHub network and calculate the persistence of them over different classes. The results bring new information about the collaborative software development process on the platform.

KEYWORDS

Social Networks, GitHub, Social Coding, Tie Strength

1 INTRODUÇÃO

A popularização e a expansão da Internet nas últimas décadas possibilitou à população em geral o acesso a uma gama de serviços e produtos, cujo uso gera grandes volumes de dados e informações que podem ser utilizados para diversas finalidades. Entre os serviços online mais populares estão as redes sociais, que conectam pessoas a partir de seus relacionamentos pessoais e profissionais [6].

Neste trabalho, estudamos o GitHub, uma plataforma online de desenvolvimento de software que pode ser visualizada como uma rede social na qual desenvolvedores são participantes que criam/contribuem/compartilham repositórios e projetos de software [1, 3, 10]. Buscamos, a partir dos dados desta plataforma, extrair informações sobre os relacionamentos entre seus usuários. Conhecer estes relacionamentos pode ser útil para diferentes aplicações, como estudo de detecção de comunidades, formação de equipes, algoritmos de recomendação, entre outros [8, 11]. Além disso, este estudo nos ajuda a compreender como desenvolvedores tendem a colaborar entre si. Sendo a colaboração essencial à engenharia de software, tal entendimento é de fundamental importância. De fato, a colaboração e o intercâmbio social influenciam o sucesso e a produtividade de uma equipe [17].

Para tais análises, é comum modelar a rede como um grafo, onde nós representam indivíduos e arestas seus relacionamentos. Também é possível considerar *aspectos temporais* das relações, pela definição do tempo em que começam e terminam. O grafo e as conexões temporais possibilitam identificar como pessoas interagem e como os relacionamentos evoluem. Dessa forma, podemos analisar a relação entre dois nós da rede a partir de suas características semânticas e topológicas e sua evolução com o tempo. Torna-se então possível inferir quais são os padrões de colaboração/interação entre os usuários da plataforma.

As contribuições deste artigo são assim resumidas. Primeiro, atualizamos uma base de dados do GitHub, inicialmente com dados até maio de 2017 [14], com dados até junho de 2019 (Seção 3.1). Em seguida, a partir dos dados atualizados, aplicamos um algoritmo de classificação de relacionamentos que considera o aspecto temporal da rede (Seção 3.3). Finalmente, avaliamos a persistência de tais relacionamentos através do tempo (Seção 3.4). De modo geral, a análise da classificação e da persistência dos relacionamentos entre os desenvolvedores definem novidades em relação ao entendimento do processo colaborativo de desenvolvimento de software na plataforma que podem ser utilizadas para predizer novas interações.

2 TRABALHOS RELACIONADOS

As redes sociais são objetos de análises de diferentes estudos, como a formação de comunidades [16] e a detecção de indivíduos influentes [2, 18]. Usualmente, redes sociais são dinâmicas e mudam com o tempo. Por exemplo, relacionamentos entre indivíduos são feitos e desfeitos de um momento para o outro. Por isso, o aspecto temporal tem sido considerado em muitos dos trabalhos recentes voltados para seus estudos [12, 21]. Redes sociais como o GitHub, i.e. redes de desenvolvimento colaborativo de software, têm sido exploradas para diversas finalidades. Por exemplo, para entender o que influencia um relacionamento entre desenvolvedores [7, 20], e identificar desenvolvedores e repositórios influentes [15, 19].

Especificamente, Alves et al. [1] definem a rede de colaboração do GitHub como um grafo não-direcionado $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, onde $\mathcal{V}$ é o conjunto de nós que representam os desenvolvedores e $\mathcal{E}$ é o conjunto de arestas que lhes conectam quando ambos colaboram em um mesmo repositório, sendo o peso das arestas calculado por diferentes métricas. Outros trabalhos utilizam tal modelagem para propor métricas e analisar os relacionamentos no GitHub de diversas formas [4, 14]. Entretanto, nenhuma dessas análises considera o aspecto temporal e a persistência dos relacionamentos na rede ao longo do tempo. Como o GitHub é uma rede dinâmica que evolui com o tempo, tal aspecto é de fundamental importância. Nesse sentido, este trabalho visa analisar os relacionamentos na plataforma do ponto de vista temporal.

In: XVII Workshop de Trabalhos de Iniciação Científica (WTIC 2020), São Luís, Brasil. Anais Estendidos do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia). Porto Alegre: Sociedade Brasileira de Computação, 2020.

ISSN 2596-1683

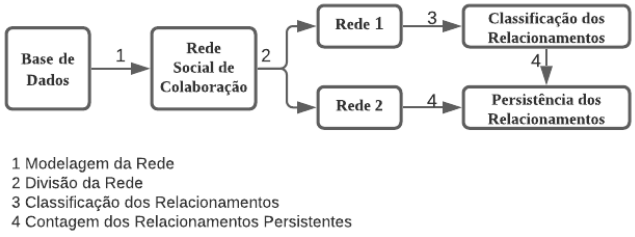


Figure 1: Metodologia proposta para este trabalho.

3 METODOLOGIA

Esta seção descreve a base de dados (Seção 3.1), a modelagem da rede social (Seção 3.2), os algoritmos usados para classificar as arestas (Seção 3.3) e o cálculo da persistência dos relacionamentos (Seção 3.4). A Figura 1 sumariza os principais passos seguidos.

3.1 Base de Dados

A base de dados utilizada é proveniente do GitSED *(GitHub Socially Enhanced Dataset)* [3], um conjunto de dados do GitHub curado, expandido e enriquecido a partir do GHTorrent [9]. O GitSED possui informações sobre usuários e repositórios, além de métricas topológicas e semânticas para a rede de colaboração do GitHub. Sua última versão contém dados até maio de 2017 e considera seis linguagens de programação subdivididas em dois grupos: linguagens mais colaborativas (JavaScript, Ruby e Python) e linguagens menos colaborativas (Assembly, Pascal e Visual Basic) [14].

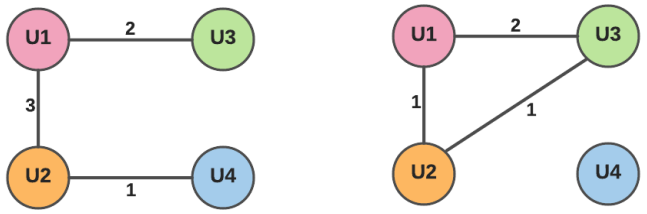
Neste trabalho, utilizamos a mesma metodologia de coleta e atualizamos a base com dados do GHTorrent até Junho de 2019. Mantivemos a subdivisão da rede entre grupos de linguagens a fim de comparar o comportamento dos grupos diante das aplicações propostas. A manutenção da fonte e a atualização do dataset com dados mais recentes permite uma avaliação da evolução do GitHub ao longo dos anos. Dessa forma, é possível dizer se o comportamento dos desenvolvedores e seu padrão de colaboração sofreram alterações, sendo esta uma das contribuições deste trabalho.

3.2 Modelagem da Rede

Para modelar a rede de colaboração entre desenvolvedores do GitHub, utilizamos como base a rede $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ da Seção 2. Desta rede, consideramos apenas repositórios originais, isto é, repositórios que não foram gerados por *fork*[1] de outros, a fim de evitar relacionamentos duplicados [3]. Para avaliar o desenvolvimento temporal dos relacionamentos no GitHub, particionamos a rede base em duas, uma para aplicação do algoritmo de classificação (Rede 1) e outra para o cálculo da persistência de arestas (Rede 2).

Para particionar a rede, selecionamos, para cada linguagem de programação, as datas do primeiro e do último *commit* e calculamos a janela de tempo que divide o período formado por elas na proporção 80/20, seguindo a mesma metodologia apresentada em [5]. Dessa forma, os pares da rede base com interações iniciadas até a janela de tempo foram selecionados para a Rede 1 (chamamos de passado) e o os pares da rede base com interações finalizadas após a janela de tempo foram selecionados para a Rede 2 (chamamos de futuro). Adicionamos um peso nas arestas da Rede 1, definido como o número de repositórios compartilhados, até a janela de tempo, entre os desenvolvedores (SR - Shared Repositories). Esta métrica, que foi proposta por Alves et al. [1], representa a intensidade de um relacionamento, indicando em quantos projetos diferentes dois desenvolvedores trabalharam juntos.

[1] *Fork*: cópia de um repositório para que mudanças sejam feitas no código sem alterar a versão original.



(a) Rede 1 (Passado) **(b) Rede 2 (Futuro)**

Figure 2: Exemplo de rede de colaboração do GitHub particionada temporalmente entre (a) passado e (b) futuro.

Table 1: Classes de relacionamentos do STACY.

Classe	Persistência das arestas	Sobreposição de vizinhos	Frequência de interação
strong	social	social	social
bridge+	social	random	social
transient	random	social	social
periodic	social	social	random
bursty	random	random	social
bridge	social	random	random
weak	random	social	random
random	random	random	random

Um exemplo de rede, dividido em duas janelas temporais, é apresentado nas Figuras 2a e 2b, que representam o passado e o futuro, respectivamente. Na Figura 2a, observa-se que os usuários 1 e 2 colaboraram juntos em três repositórios, os usuários 1 e 3 em dois repositórios e, por fim, os usuários 2 e 4 em um único repositório. Já na Figura 2b, temos que o par de usuários 1 e 2 e o par de usuários 2 e 3 tiveram interações em um único repositório, o par 1 e 3 em dois repositórios e o usuário 4 não interagiu com qualquer outro. Ao analisar a evolução dos relacionamentos com o tempo, vemos que os relacionamentos U1-U2 e U1-U3 persistiram, enquanto o relacionamento U2-U3 deixou de existir. Nesse exemplo, o peso das arestas é significante à continuidade da interação. Neste trabalho, o objetivo é investigar quais características (representadas pelo peso da aresta) das relações são fatores importantes à persistência.

3.3 Classificação da Força dos Relacionamentos

Para classificar os relacionamentos entre os desenvolvedores, utilizamos o STACY (*Strength of Ties Automatic-Classifier over the Years*) [5]. O STACY classifica a força entre as arestas da rede considerando o aspecto temporal e: (i) a persistência das arestas, (ii) a sobreposição de vizinhos e (iii) a frequência da interação (peso das arestas). Assim, dependendo da força, o relacionamento é atribuído a uma das oito classes apresentadas na Tabela 1.

O algoritmo considera o aspecto temporal das interações ao dividir as entradas em intervalos de tempo previamente definidos e analisa as variáveis (persistência, sobreposição, frequência) ao longo destes intervalos. Neste trabalho, sabendo que uma semana é um período de tempo considerado relevante em desenvolvimento de software [13], utilizamos intervalos de sete dias.

3.4 Persistência dos Relacionamentos

Para calcular a persistência dos relacionamentos, contamos, para cada classe gerada pelo STACY, a porcentagem de arestas que persistiram na Rede 2. Especificamente, o algoritmo STACY é aplicado no passado, e então, é verificado se as arestas de cada classe (**strong**, **bridge+**, **transient**, **periodic**, **bursty**, **bridge**, **weak** e **random**) continuam na mesma classe no futuro.

4 RESULTADOS

Esta seção apresenta uma caracterização das redes de colaboração do GitHub (Seção 4.1) e uma análise dos relacionamentos nessas redes (Seção 4.2). Ademais, são apresentados os resultados para as linguagens Assembly e JavaScript, que possuem baixo e alto nível de colaboração, respectivamente. Tais linguagens foram escolhidas como representantes, pois outras linguagens do mesmo grupo possuem comportamento similar [14].

4.1 Caracterização das Redes de Colaboração

A Tabela 2 apresenta uma caracterização das redes das linguagens analisadas neste trabalho, incluindo: número de repositórios, número de nós, número de pares de desenvolvedores conectados, densidade[2], grau médio dos nós[3], número de nós na *Giant Component* (GC)[4] e número de arestas na GC. Nota-se que a maioria das arestas da linguagem mais colaborativa estão na GC, o que mostra que tal rede é bem conectada. Por outro lado, a rede menos colaborativa possui grau médio de nós baixo e uma baixa porcentagem de nós e arestas na GC. Conseqüentemente, podemos dizer que os repositórios dessa linguagem são compostos, em sua maioria, por poucos desenvolvedores que pouco colaboram entre si, corroborando a definição de colaboração. É importante enfatizar que a atualização dos dados do GitSED mostra que não houve alteração no comportamento dos desenvolvedores dessas linguagens, visto que JavaScript e Assembly permanencem como colaborativa e não colaborativa, respectivamente. Por fim, a Tabela 3 apresenta a quantidade de nós e arestas para as duas divisões das redes de colaboração das linguagens Assembly e JavaScript.

4.2 Análise dos Relacionamentos nas Redes de Colaboração do GitHub

Nesta seção, a análise dos relacionamentos é feita considerando duas abordagens, a classificação da força dos relacionamentos e a análise de como eles persistem ao longo do tempo.

Classificação da força dos relacionamentos. A Figura 3 mostra que a maioria dos relacionamentos são classificados como fracos

[2] *Densidade:* porcentagem do número de arestas em um grafo completo com o mesmo número de nós. Seu valor é 0 para um grafo sem arestas e 1 para um grafo completo.
[3] *Grau de um nó:* número de arestas conectadas a ele.
[4] *Giant Component*: o maior, em número de nós, componente conectado de um grafo.

Table 2: Estatísticas das Redes Assembly e JavaScript.

Descrição	Assembly	JavaScript
Repositórios	3.245	615.300
Usuários (nós)	9.353	1.152.467
Pares (arestas)	18.843	5.262.491
Densidade (10^{-3})	0,4	0,008
Grau médio dos nós	4,03	9,13
Nós no GC	537 (5,74%)	499.588 (43,35%)
Arestas no GC	4.124 (21,89%)	4.634.352 (88,06%)

Table 3: Divisão das Redes Assembly e JavaScript.

Descrição	Assembly		JavaScript	
	Rede 1	Rede 2	Rede 1	Rede 2
Nós	6.413	284	736.421	42.884
Arestas	12.549	593	2.328.487	158.457

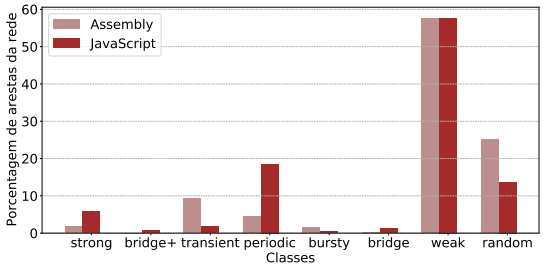


Figure 3: Classificação da força dos relacionamentos.

(frequência de interação baixa, persistência de interação baixa e sobreposição de vizinhança alta), independente do grau de colaboração da linguagem. Uma possível explicação para isso é que o comportamento padrão do usuário do GitHub é o de realizar uma contribuição breve a um repositório específico. Além disso, a linguagem mais colaborativa possui maior proporção de relacionamentos *strong* (frequência de interação alta, persistência de interação alta e sobreposição de vizinhança alta) e *periodic* (frequência de interação baixa, persistência de interação alta e sobreposição de vizinhança alta). Esse resultado indica que linguagens mais colaborativas tendem a ter mais desenvolvedores trabalhando juntos em repositórios do que pares de desenvolvedores com valores de SR mais altos. Por outro lado, a linguagem menos colaborativa possui maior proporção de relacionamentos *random* (frequência de interação baixa, persistência de interação baixa e sobreposição de vizinhança baixa) e *transient* (frequência de interação alta, persistência de interação baixa e sobreposição de vizinhança alta). Essas classes compartilham uma baixa persistência de interação, o que indica que as trocas entre desenvolvedores de linguagens menos colaborativas tendem a ser isoladas.

Já a Tabela 4 indica a quantidade média de repositórios dos usuários pertencentes a cada classe. Em ambas as linguagens, nota-se que os desenvolvedores das classes *strong*, *transient* e *periodic*

Table 4: Média de repositórios por usuário em cada classe.

	Assembly	JavaScript
strong	61,75	42,06
bridge+	27,2	25,64
transient	36,4	54,87
periodic	40,15	29,57
bursty	19	27,27
bridge	7,15	18,85
weak	9,96	14,68
random	3,37	6,62

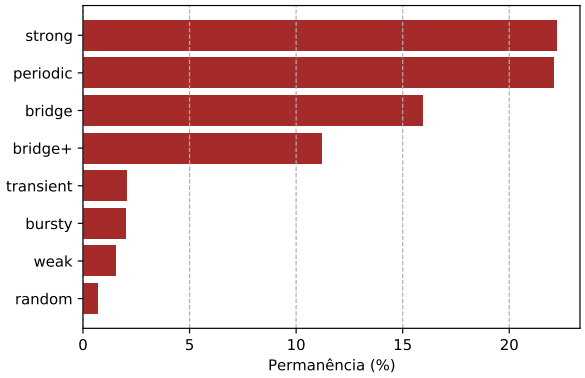


Figure 4: Persistência de arestas por classe.

são os que possuem a maior média de repositórios. Tais classes possuem em comum uma alta sobreposição de vizinhos, o que pode indicar que os desenvolvedores em questão estão em uma parte bastante conectada na rede, i.e., com alta diversidade nas colaborações. Em geral, observa-se que desenvolvedores das classes de relacionamento mais fortes possuem mais repositórios, de modo que possa indicar uma correlação entre relacionamentos fortes e duradouros e a produtividade dos usuários que os compõem.

Persistência dos relacionamentos por classe. A Figura 4 apresenta a porcentagem de arestas persistentes, por classe, para a rede JavaScript. Os resultados obtidos para a rede Assembly foram similares. Observa-se que as classes de maior persistência de arestas (*strong*, *periodic* e *bridges*) são as que mais se mantiveram no "futuro", o que mostra que a classificação pode ser interpretada como previsão para interações futuras. Quase 1/4 das arestas classificadas como *strong* e *periodic* persistem. Isso mostra que a sobreposição de vizinhanças também é um fator importante para a persistência. Esses resultados indicam que as métricas utilizadas pelo STACY podem ser aplicadas na predição de *links* para o GitHub.

5 CONCLUSÃO

Neste trabalho, foi investigado o comportamento dos relacionamentos entre desenvolvedores no GitHub. Para isso, foi realizada a classificação da força dos relacionamentos, considerando o aspecto temporal das interações dos usuários em repositórios, e também foi analisada a persistência desses relacionamentos ao longo do tempo. As análises revelaram que a maioria dos relacionamentos são classificados como fracos, o que indica que os usuários tendem a realizar uma contribuição breve a um repositório específico. Foi possível concluir que a classificação pode auxiliar na previsão de interações futuras, ao mostrar que as arestas mais fortes tendem a persistir mais. Finalmente, a análise da persistência dos relacionamentos indicou a relevância da sobreposição de vizinhos nesse aspecto. Como trabalho futuro, pretende-se desenvolver um algoritmo de predição de *links* para recomendação de usuários para o GitHub.

ACKNOWLEDGMENTS

Esse trabalho foi parcialmente financiado pelo CNPq.

REFERENCES

[1] Gabriela B. Alves et al. 2016. The Strength of Social Coding Collaboration on GitHub. In *SBBD*. SBC, 247–252.
[2] Carlos V. S. Araújo, Rayol M. Neto, Fabíola Guerra Nakamura, and Eduardo Freire Nakamura. 2017. Using Complex Networks to Assess Collaboration in Rap Music: A Study Case of DJ Khaled. In *WebMedia*. ACM, 425–428.
[3] Natércia A. Batista et al. 2017. Collaboration strength metrics and analyses on GitHub. In *WI*. ACM, 170–178.
[4] Natércia A. Batista, Guilherme A. de Sousa, Michele A. Brandão, Ana Paula Couto da Silva, and Mirella Moura Moro. 2018. Tie Strength Metrics to Rank Pairs of Developers from GitHub. *J. Inf. Data Manag.* 9, 1 (2018), 69–83. https://periodicos.ufmg.br/index.php/jidm/article/view/417
[5] Michele A Brandão, Pedro OS Vaz de Melo, and Mirella M Moro. 2018. STACY: strength of ties automatic-classifier over the years. *Journal of Information and Data Management* 9, 1 (2018), 52–52.
[6] Michele A. Brandão and Mirella M. Moro. 2017. Social professional networks: A survey and taxonomy. *Comput. Commun.* 100 (2017), 20–31.
[7] Casey Casalnuovo, Bogdan Vasilescu, Premkumar T. Devanbu, and Vladimir Filkov. 2015. Developer onboarding in GitHub: the role of prior social links and language experience. In *ESEC/SIGSOFT FSE*. ACM, 817–828.
[8] Laura A. Dabbish, H. Colleen Stuart, Jason Tsay, and James D. Herbsleb. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. In *CSCW*. ACM, 1277–1286.
[9] Georgios Gousios. 2013. The GHTorent Dataset and Tool Suite *(MSR '13)*. IEEE Press, 233–236.
[10] Libo Li, Frank Goethals, Bart Baesens, and Monique Snoeck. 2017. Predicting software revision outcomes on GitHub using structural holes theory. *Comput. Networks* 114 (2017), 114–124.
[11] Libo Li, Frank Goethals, Bart Baesens, and Monique Snoeck. 2017. Predicting software revision outcomes on GitHub using structural holes theory. *Comput. Networks* 114 (2017), 114–124.
[12] Qiu Liqing, Yu Jinfeng, Fan Xin, Jia Wei, and Wenwen Gao. 2019. Analysis of Influence Maximization in Temporal Social Networks. *IEEE Access* 7 (2019), 42052–42062.
[13] Michael Boyer O'Leary et al. 2011. Multiple team membership: A theoretical model of its effects on productivity and learning for individuals and teams. *Academy of Management Review* 36, 3 (2011), 461–478.
[14] Gabriel P. Oliveira et al. 2018. Tie Strength in GitHub Heterogeneous Networks. In *WebMedia*. ACM, 363–370.
[15] Leiming Ren, Shimin Shan, Xiujuan Xu, and Yu Liu. 2020. StarIn: An Approach to Predict the Popularity of GitHub Repository. In *ICPCSEE (2) (Communications in Computer and Information Science)*, Vol. 1258. Springer, 258–273.
[16] Anwar Said, Rabeeh Ayaz Abbasi, Onaiza Maqbool, Ali Daud, and Naif Radi Aljohani. 2018. CC-GA: A clustering coefficient based genetic algorithm for detecting communities in social networks. *Appl. Soft Comput.* 63 (2018), 59–70.
[17] Jonathan Sillito and Eleanor Wynn. 2007. The Social Context of Software Maintenance. In *ICSM*. IEEE Computer Society, 325–334.
[18] Arlei Silva et al. 2011. From Individual Behavior to Influence Networks: A Case Study on Twitter. In *WebMedia*. SBC, 135–142.
[19] Ferdian Thung et al. 2013. Network Structure of Social Coding in GitHub. In *CSMR*. IEEE Computer Society, 323–326.
[20] Bogdan Vasilescu, Alexander Serebrenik, and Vladimir Filkov. 2015. A Data Set for Social Diversity Studies of GitHub Teams. In *MSR*. IEEE Computer Society, 514–517.
[21] Tongfeng Weng, Yi Zhao, Michael Small, and Defeng David Huang. 2014. Time-series analysis of networks: Exploring the structure with random walks. *Physical Review E.* 90, 2 (2014), 022804.